

I/O Monitoring in a Hadoop Cluster

Joshua Long
Georgia Institute of Technology

Joel Ornstein
Harvey Mudd College

Carson Wiens
Calvin College

Abstract

As data sizes for scientific computations grow larger, more of these types of computations are bottlenecked by disk input and output, rather than processing speed. Monitoring reads and writes, therefore, has become an important component of distributed computing clusters. Our team investigated several different possibilities for monitoring I/O on a Hadoop cluster, including the Splunk app for HadoopOps, Ganglia, and log file output. Each of these three methods were evaluated for compatibility, ease of use, and display. Surprisingly, despite the fact that Splunk HadoopOps is made specifically for Hadoop clusters, other monitoring programs and techniques still proved to be useful. Using our monitoring tools, we were also able to observe input and output behavior over different cluster architectures.

Environment

- Eleven-node Centos 6.5 cluster
- 1-gigabit Ethernet connections between nodes
- FDR InfiniBand 56-Gb/second between nodes
- Hadoop 2.3, YARN, Java 1.6

Background

A recent push in the high-performance computing community has created a demand for “big data” machines - computers which store and use massive amounts of unstructured information. As always, the larger a cluster grows, the more often its components will fail, and it is vital that big data operations ensure that information is being read and written to disks efficiently. Monitoring the I/O (the reads and writes to disk) is vital to maintaining the efficiency of running processes in big data machines, such as a Hadoop cluster.

A Hadoop cluster is one variety of big data machine. It uses a striped and replicated file system, which distributes the same file system over many nodes. The data is separated into 128 MB blocks and replicated three times on each node.

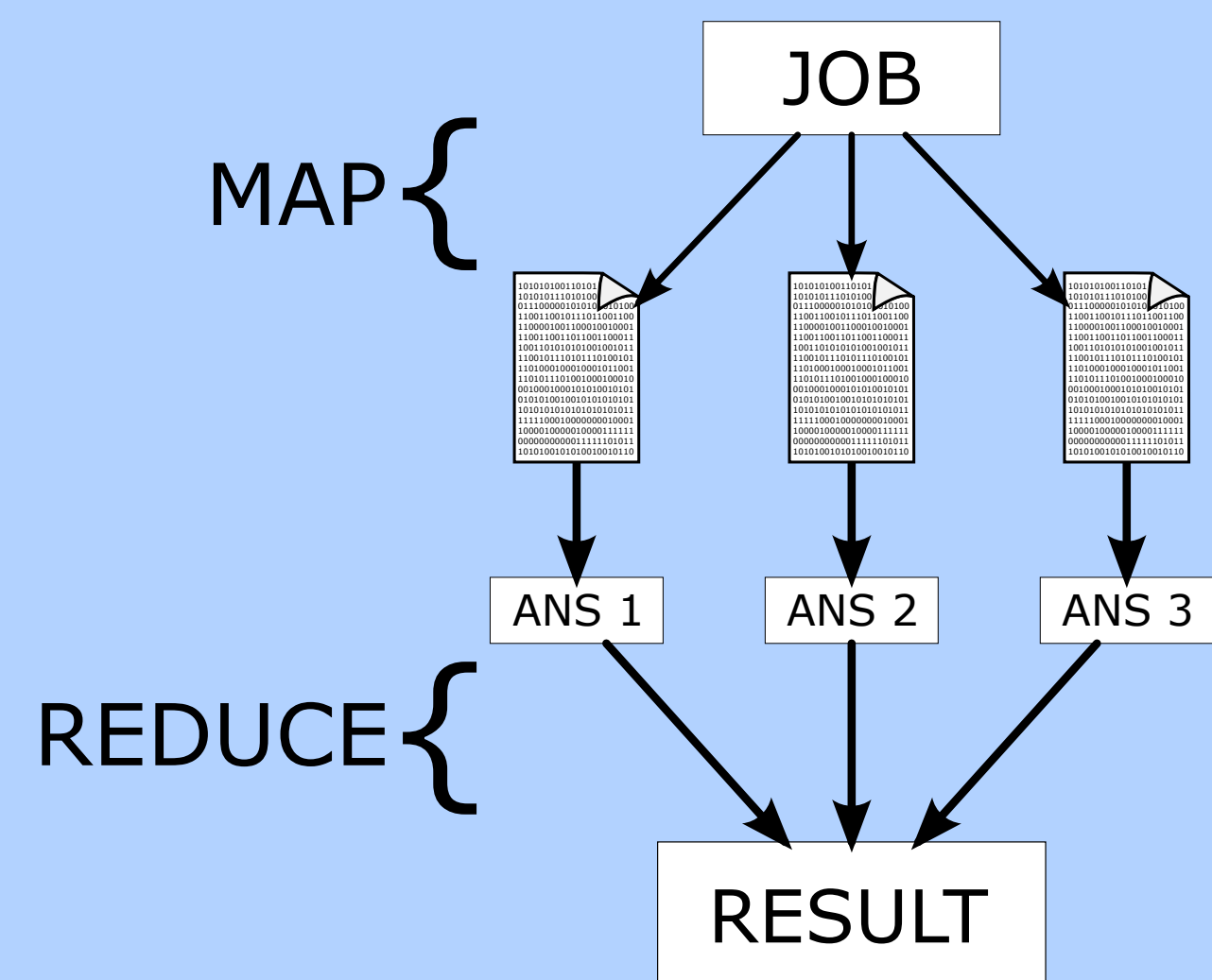


Figure 1: Hadoop is an implementation of the MapReduce algorithm which works by moving the job to the data, processing the job on different datasets in parallel, and then combining these individual answers into a single result.

The goal of the MapReduce algorithm is to perform a computation of different pieces of data and combine these intermediate answers into a consolidated result. The algorithm consists of two core steps, called map and reduce. The map step maps different processors to data to perform the computation on, which result in a set of intermediate answers. The reduce step combines the intermediate answers into a final conclusion.

Mentors: Steve Senator, Tim Randles, Vaughan Clinton, Mike Mason, Graham Van Heule
Instructor: Dane Gardner, TA: Chris Moore

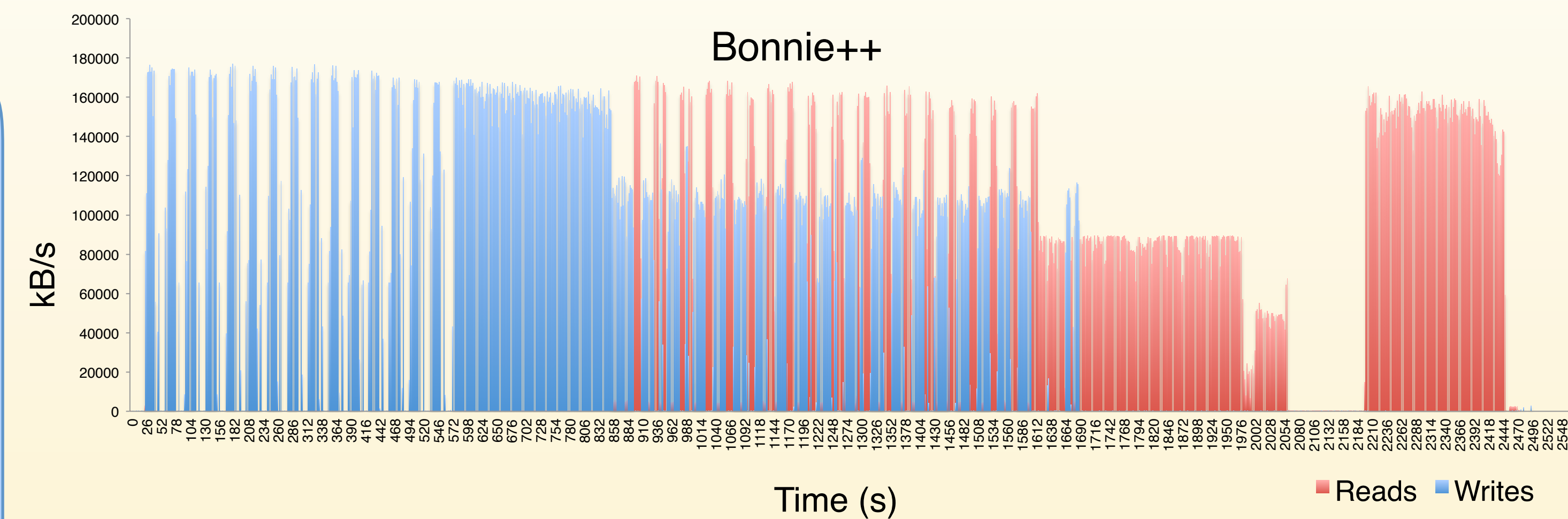


Figure 2: bonnie++ is a file system stress test. It runs several benchmarks, which read and write as quickly as the system will allow.

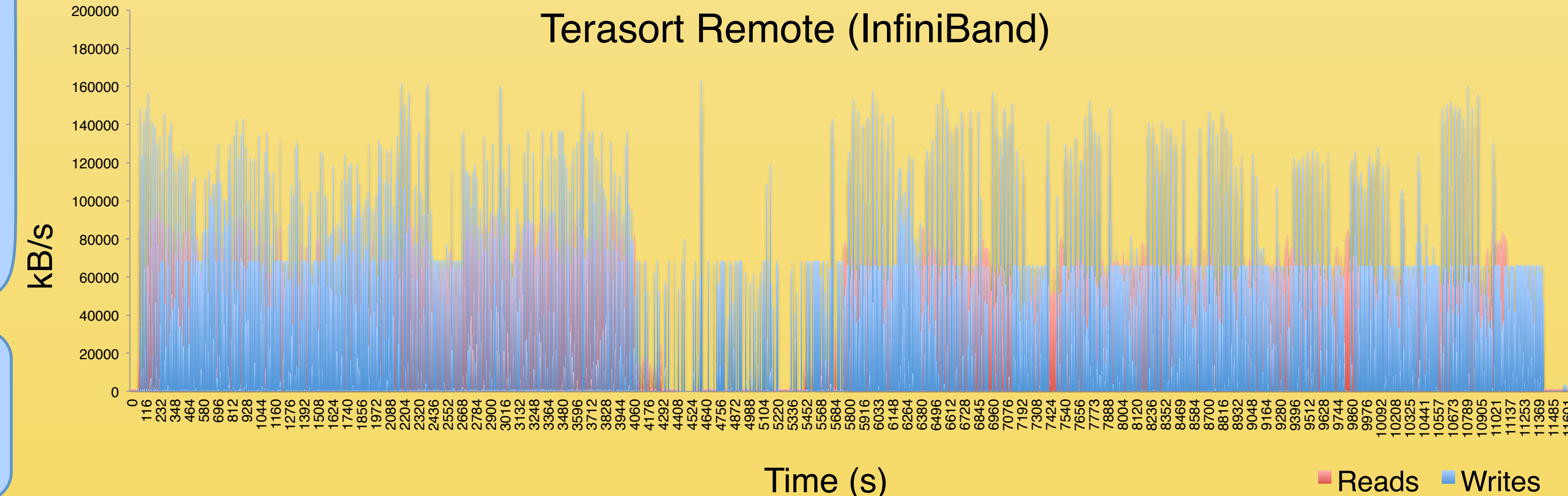
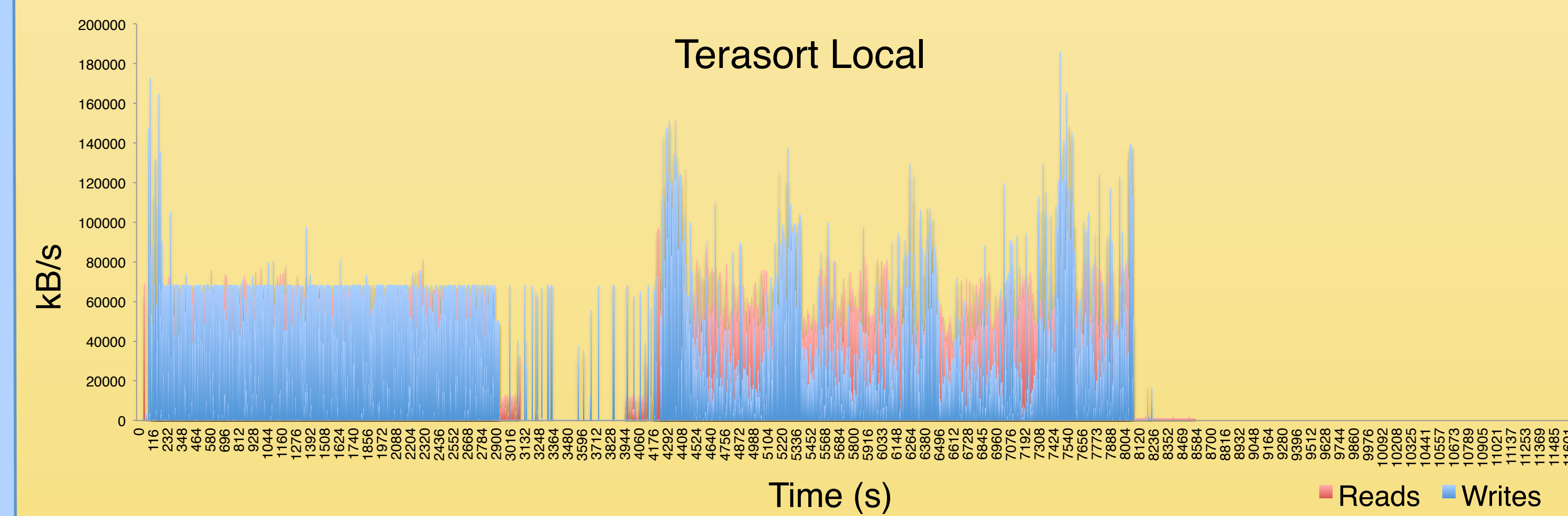
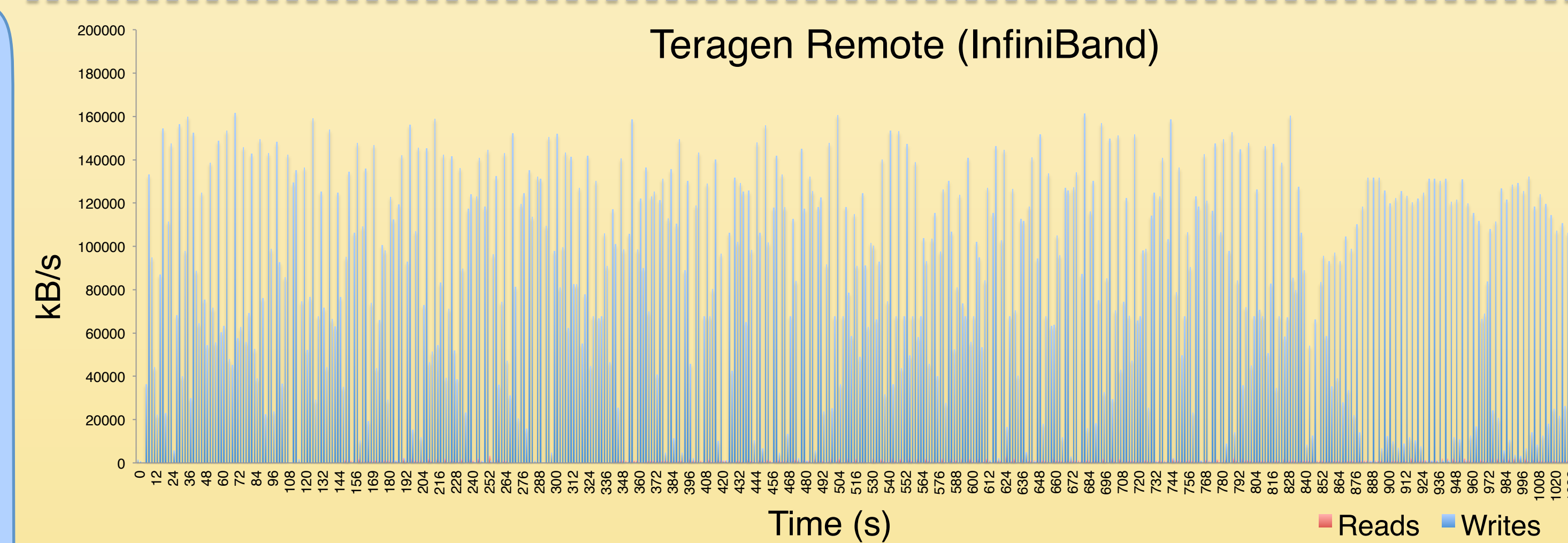
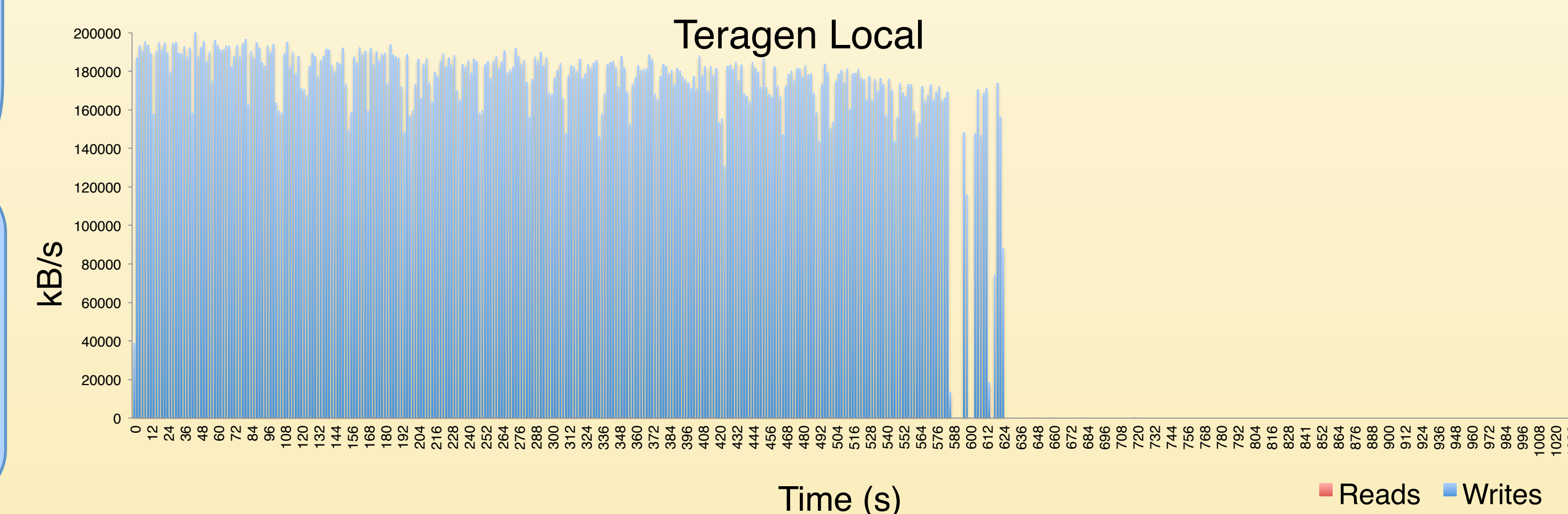


Figure 3: teragen and terasort were used to simulate running I/O-intensive jobs. We ensured that our methods of gathering metrics were correct by comparing the iostat output from the teragen and terasort jobs to bonnie++ I/O. The graphs also show the difference between running jobs with data on each node, and running jobs with data stored remotely.

Methods

Iostat is a command that reports the number of kilobytes read and written per second, the average time for I/O requests to be served, the utilization of the device, and other important I/O statistics. In order to monitor I/O during Hadoop jobs and benchmarking, we wrote a script to output statistics from iostat to a CSV log file.

We were able to successfully monitor I/O using the Ganglia web interface - an extensible monitoring program for clusters and grids. Ganglia's metric tool, gmetric, allowed us to easily specify a new metric. Our script parses iostat output and feeds it into gmetric for easy visualization on the Ganglia monitoring software.

We used bonnie++, a file-system stress

test, to benchmark I/O on the hard drives. bonnie++ is a tool that creates, measures, and deletes files in both sequential and random order while recording I/O statistics. By editing the source code, we were able to determine the beginning and ending of each of the twelve bonnie++ tests. These I/O measurements (figure 2) were used as a baseline against which we could compare data that was gathered using our iostat script.

On the cluster we ran the teragen and terasort Hadoop jobs, which we used to create and sort a terabyte of data in order to simulate an I/O-intensive job.

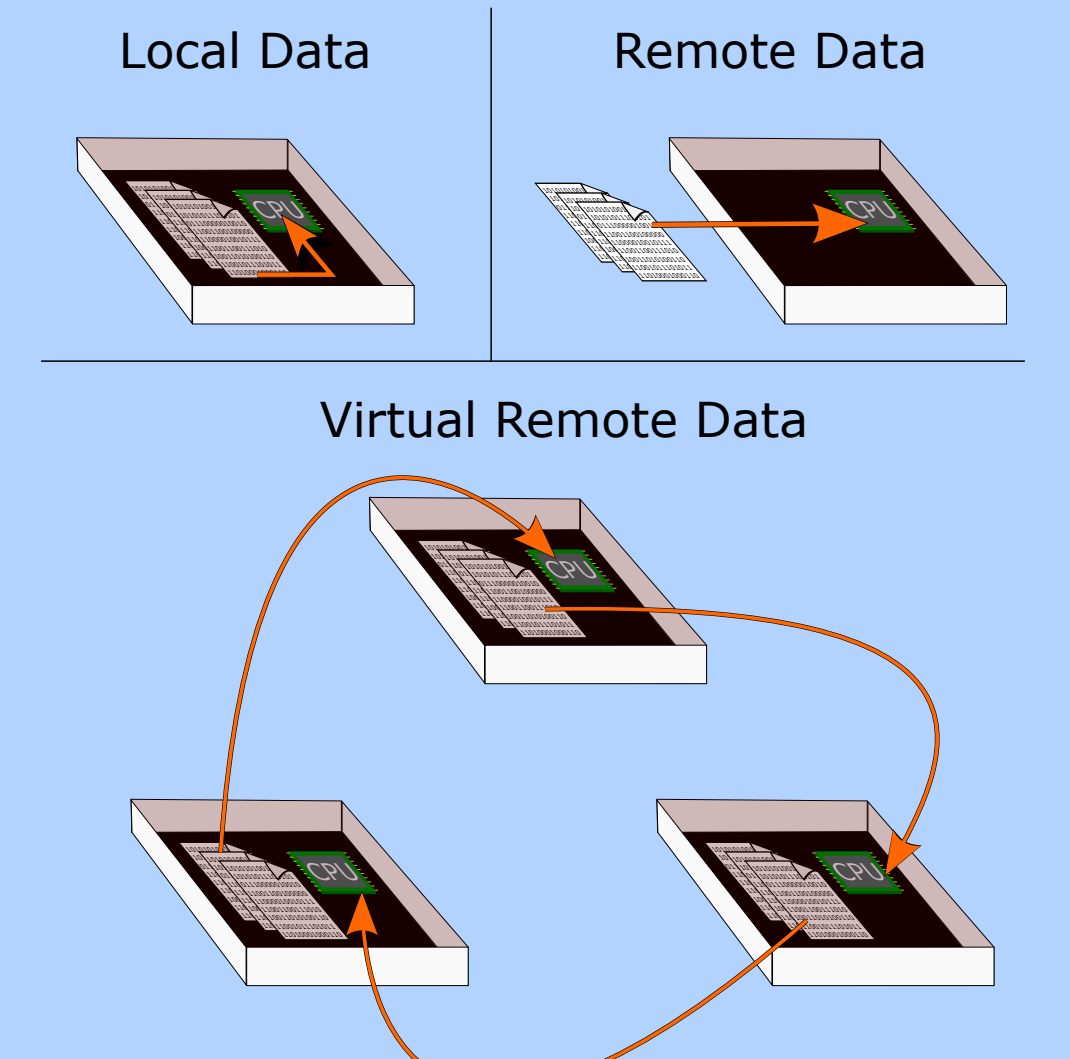


Figure 4: It is ideal for data to be hosted locally to avoid network latency. However it is not always practical to have local data. To simulate remote data, we set up our servers to serve data to each other in a ring. This virtual remote data setup was used to evaluate the effects of network latency.

Results

The Splunk App for HadoopOps is not yet ready for Hadoop cluster using MapReduce version 2, due to its ignorance of YARN, as well as its dependency on deprecated pieces of Hadoop software, such as JobTracker. In the end we were unable to coax HadoopOps into functionality. Ganglia can be easily configured to monitor I/O, although it lacks many of the features and monitoring options found in HadoopOps.

By simulating diskless clusters we found that reads and writes on remote data were usually around 60% slower than processing the same job locally, as shown by the graphs in figure 3. Since the InfiniBand connection between nodes works faster than the disks on the nodes, we know that we are getting the fastest-possible read/write times. We are able to verify that the gathered metrics are correct by comparing them to the results of our bonnie++ stress-test (figure 2), and we have simulated both local and remote YARN jobs using teragen and terasort.

Future Work

Ganglia is a quick and easy way to monitor clusters (even Hadoop clusters), however, Splunk allows for greater configuration of metrics, and it would be convenient to centralize all of a cluster's metrics into a single web interface instead of spreading out monitoring work over both Ganglia and Splunk. If we were to continue our work on monitoring, we would either extend and improve the Splunk App for HadoopOps, or write our own Splunk application in order to gather more relevant metrics for easy and configurable display. Additionally, we could also try to optimize Hadoop parameters in order to reduce the effects of network latency.

LA-UR-14-25972